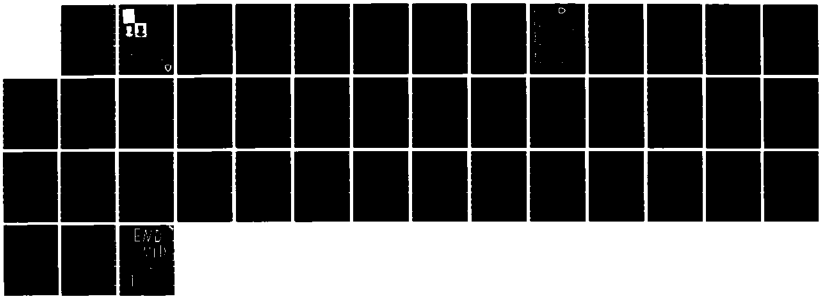
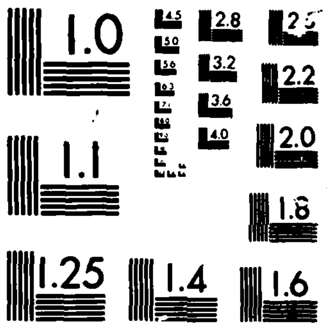
AD-A166 927 US ARMY ADA (TRADE NAME) TRAINING CURRICULUM CURRICULUM 1/1
CATALOG 1986(U) SOFTECH INC WALTHAM MA 1986
DAAB07-83-C-K514

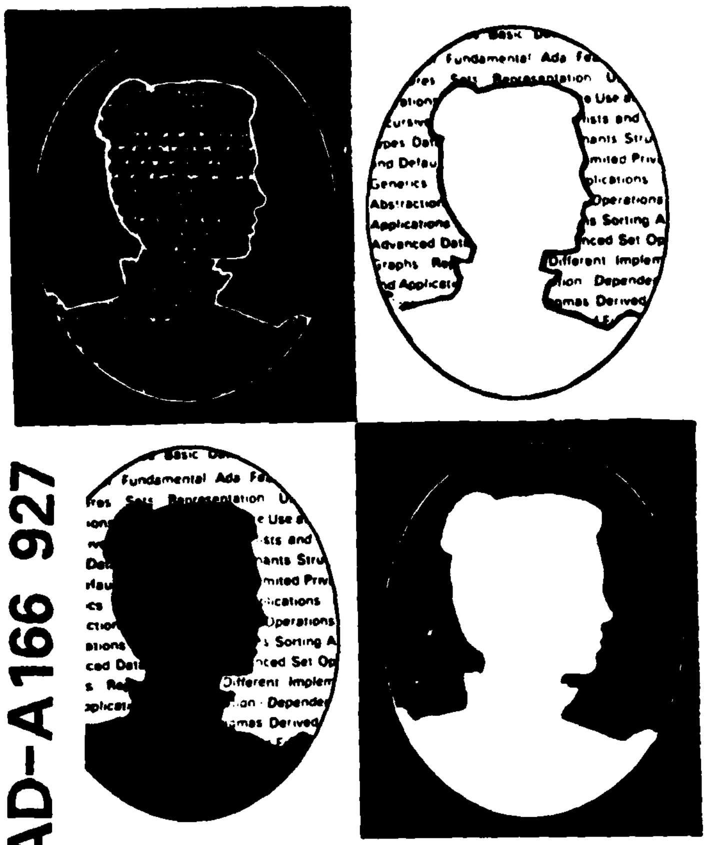
UNCLASSIFIED F/G 5/9 NL





MICROCOP' CHART

1



Superseded AD-A144188

U.S. Army Ada® Training Curriculum

Curriculum Catalog 1986

*Ada is a registered trademark of the U.S. Government, Ada Joint Program Office. The U.S. Army Ada Training Curriculum was developed by SofTech, Inc. under the Ada Design Methods Training Support contract (DAAB07-83-C-K514) sponsored by the Software Technology Development Division (CENTACS) of the U.S. Army Communications Electronics Command (CECOM), Fort Monmouth, New Jesey. APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

USING THE MODULAR ADA CURRICULUM

Background

The Ada Training Curriculum defines a comprehensive set of training course modules or building blocks which can be connected in a variety of ways and coupled with workbooks and supplementary materials to form training programs that satisfy a given set of needs.

A complete list of the course modules is given in Table I (p.5) and Table II (p.6) lists the workbooks and supplementary materials. The course modules differ in one or more of the following dimensions.

1. Area. It is recognized that knowledge of a programming language cannot be effectively separated from knowledge of a software engineering methodology and the tools that support it (the software "environment"). Modules whose identifier starts with the letter L are concerned with the Ada language proper; the letters M and E identify, respectively, methodology and environment modules.

2. Depth. The curriculum is designed to avoid the need for training every individual in every aspect of the language, methodology or environment. In each module identifier, the initial letter is followed by a digit expressing the module level. A level-1 module is one having no prerequisites or general prerequisites not related to Ada (for example, one such prerequisite might be the knowledge of some high-level language). Level-2 modules have prerequisites that can be satisfied by level-1 modules and so on.

3. Viewpoint. Ada and Software Engineering are of interest to more than just programmers. Top-level managers need to know concepts of a technology but may have little need for the "nuts and bolts" of a technology. Thus there are basically two viewpoints addressed by the curriculum: a practitioner's (technical) viewpoint and a manager's viewpoint.

To provide the necessary flexibility the curriculum does not prescribe any of the following:

a. The teaching methodology, presentation techniques and media used in each course module.

b. The exact training required by each individual of an organization and the total set of skills to be taught in an organization

c. The "packaging" of course modules into training programs. It should be remembered that a course module defines a capsule of knowledge not necessarily a complete course. The most effective training programs will be those integrating several course modules possibly from different areas.

The curriculum does define a set of precedences among the course modules as shown in Figure 1 (p.7) and listed in the course module descriptions. The intended interpretation of Figure 1 is as follows: Inputs to the box corresponding to a given course module define the prerequisites for that module. It is specifically not the intent to recommend specific paths through the chart. In other words a line from a box B1 to a box B2 means that module B2, if of interest, should be taken after module B1; it does not mean that after taking module B1 an individual must proceed to module B2.

Packaging

Selecting a training plan, even a partial one, is an operation requiring careful consideration. The most common mistake is to treat the curriculum as a linear menu looking for "the" one complete course (consisting of several course modules) that will satisfy the needs of a certain audience. To use an analogy, that is like scanning a computer manufacturer's catalog in search of "the" component that will satisfy a user's requirement. The modern practice is to offer a variety of interoperable CPU's, memories, terminals, and so on; to assemble a system one must understand the function of each device.

Continuing with the previous analogy, it is important not to confuse the CPU with "the computer". A competent salesperson strives to provide the customers with a capability that is (a) complete, (b) well-balanced, and (c) appropriate for the intended usage. Packaging of course modules is not too hard once the basic principles are understood.

1. Define the viewpoint. As was previously stated, there are two main viewpoints addressed by the curriculum: one could be called the practitioner's (technical) viewpoint and is aimed at people who will actually write Ada code. Those course modules with a triangle in Figure 1 address practitioners. The other viewpoint might be called the manager's viewpoint and applies to anyone who does not need actual working knowledge of Ada. These course modules are labeled with a circle in Figure 1. Manager's courses tend to be shorter and concept-oriented; it is definitely wrong to assume that managerial courses are superficial; in many cases the emphasis on concepts (as opposed to details) makes managerial courses deeper than any practitioner's courses. For example managerial courses are very well suited for contract monitors and people doing in-depth QA.

2. Define the level. For practitioner courses, the 100 series generally includes introductory courses, intended mostly as prerequisites for other courses; higher series indicate more and more advanced courses. For managerial courses the 100 series often includes a very high-level overview adequate for top management; higher series are generally appropriate for software managers and other people who influence the software without writing code (QA, internal consultants, analysts and so on).

3. Identify the main courses. Having defined viewpoint and level it is easy to pick what sounds like the best course. The most popular candidates for programmers are Basic Ada Programming (L202), Advanced Ada Topics (L305), Real-Time Systems In Ada (L401) and Software Engineering Methodologies (M201). For software managers, analysts, program monitors and senior QA personnel, Ada For Software Managers (L201) is appropriate. For top-level managers Ada Orientation For Managers (L101) is appropriate.

4. Search for related courses. It is very important to understand that a language course without parallel training in methodology and software tools (environment) is pretty much like a CPU without memory. The only reason why the curriculum includes stand-alone language courses is that different organizations use different methodologies and different environments. (In theory it is also possible to have to train an organization that is well-versed in software engineering, but in the context of some other high-level language.)

   In particular, the following courses are virtually indivisible: (1) L101 and M101; (2) L102 and M102; (3) L202 and M203. Ideally for each of these pairs an environment course should be added. Currently, there are few environment courses; therefore courses like L202 are supplemented with a brief introduction to the basic tools needed to develop exercise programs (editor-VAX/VMS Editor; and compiler Ada/Ed translator).

   Depending on the preferred training style, the methodology course can precede the language course or be taught in parallel; the first approach is more appropriate for intense, five-days-per-week formats. The parallel format is preferable with full-semester formats.

5. Don't forget the prerequisites! The most common mistake in packaging is to focus on the "meaty" courses (L201, L202, L305,...) forgetting that all such courses have prerequisites. Unless there is assurance that the students already have the necessary prerequisites, it is generally necessary to include a few low-end series course modules in the training package. Figure 1 indicates the prerequisites very clearly.

As is always the case, this scheme has a few slight exceptions. First, M203 really works best if taught immediately after or in the middle of L202. Second, the prerequisites need not be taken strictly in terms of the curriculum. Exactly how these prerequisites have been acquired is not too important. What is important is that a student have the necessary background to take a given course module.

TABLE I

ADA LANGUAGE COURSE MODULES

Module Number	Module Name	Page No.
L101	Ada Orientation For Managers	9
L102	Ada Technical Overview	10
L103	Introduction To Ada-A Higher Order Language	11
L201	Ada For Software Managers	12
L202	Basic Ada Programming	14
L303	Real-Time Concepts	16
L305	Advanced Ada Topics	17
L401	Real-Time Systems In Ada	19
L402	Using The Ada Language Reference Manual	21

METHODOLOGY COURSE MODULES

Module Number	Module Name	Page No.
M101	Software Engineering For Managers	24
M102	Introduction To Software Engineering	25
M201	Software Engineering Methodologies	26
M203	Programming Methodology	27

ADA LANGUAGE SYSTEM (ALS) COURSES

Course Number	Course Name	Page No.
E100	Ada Language System (ALS) User Course	29
E200	Ada Language System (ALS) Administrator Course	31

TABLE II

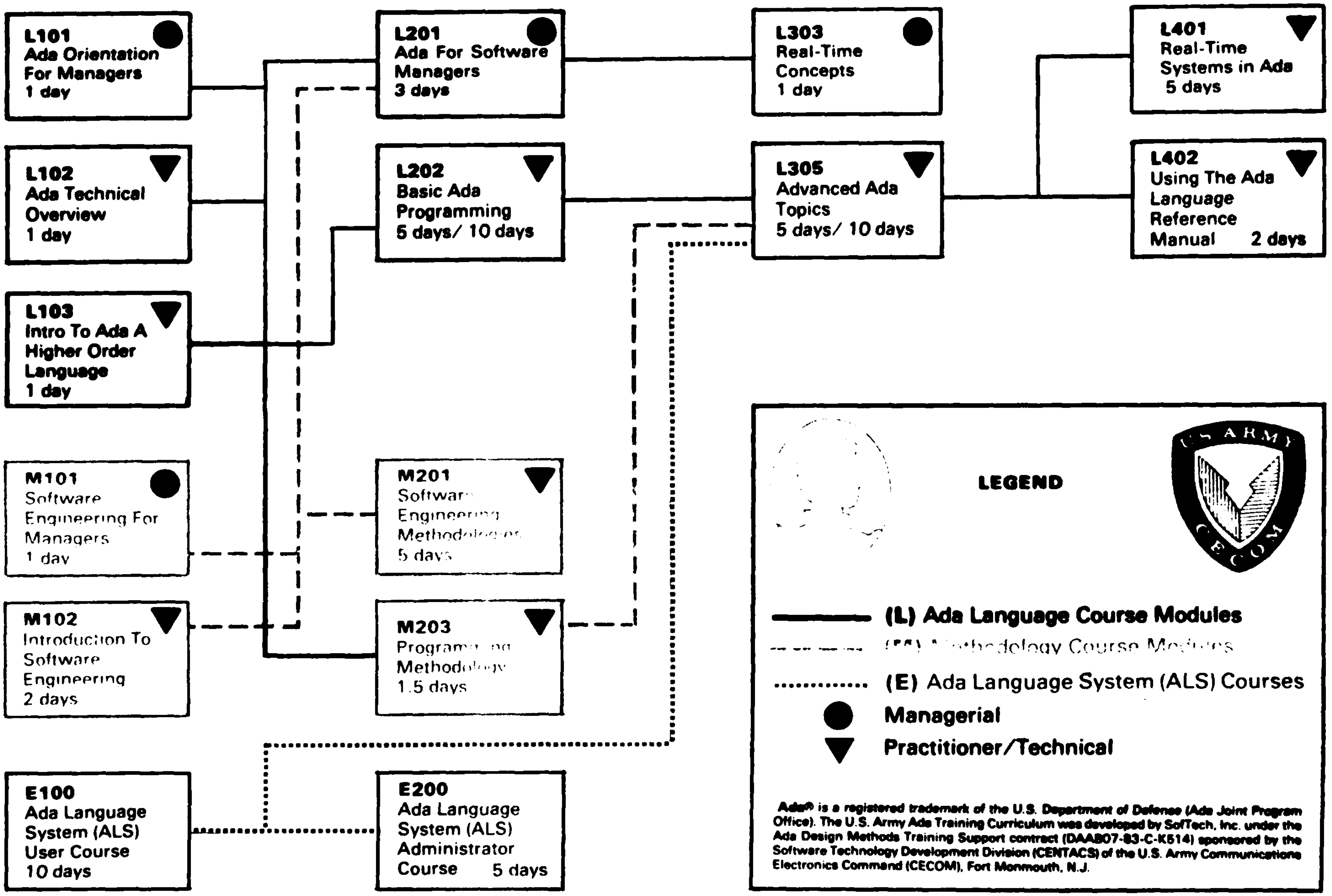
ADA LANGUAGE WORKBOOKS

Workbook Title	Corresponding Module	Page No.
Ada Primer	L202	33
Advanced Ada	L305	34
Real-Time Ada	L401	35

SUPPLEMENTARY MATERIAL

Title	Page No.
Case Studies Report	37
Ada Case Studies II	38
Ada Language System (ALS) Textbook	39
Using Selected Features of Ada: A Collection of Papers	40
Designing Real Time Systems in Ada	41

U.S. ARMY Ada® TRAINING CURRICULUM


L101 Ada Orientation For Managers 1 day ●
L102 Ada Technical Overview 1 day ▼
L103 Intro To Ada A Higher Order Language 1 day ▼
M101 Software Engineering For Managers 1 day ●
M102 Introduction To Software Engineering 2 days ▼
E100 Ada Language System (ALS) User Course 10 days
L201 Ada For Software Managers 3 days ●
L202 Basic Ada Programming 5 days/ 10 days ▼
M201 Software Engineering Methodologies 5 days ▼
M203 Programming Methodology 1.5 days ▼
E200 Ada Language System (ALS) Administrator Course 5 days
L303 Real-Time Concepts 1 day ●
L305 Advanced Ada Topics 5 days/ 10 days ▼
L401 Real-Time Systems in Ada 5 days ▼
L402 Using The Ada Language Reference Manual 2 days ▼
US ARMY CECOM
LEGEND
(L) Ada Language Course Modules
(M) Methodology Course Modules
(E) Ada Language System (ALS) Courses
● Managerial
▼ Practitioner/Technical

Ada® is a registered trademark of the U.S. Department of Defense (Ada Joint Program Office). The U.S. Army Ada Training Curriculum was developed by SofTech, Inc. under the Ada Design Methods Training Support contract (DAAB07-83-C-K514) sponsored by the Software Technology Development Division (CENTACS) of the U.S. Army Communications Electronics Command (CECOM), Fort Monmouth, N.J.

ADA LANGUAGE COURSE MODULES

NAME:	Ada Orientation For Managers (L101)
DURATION:	1 day
OBJECTIVE:	To give managers an overview of the development and features of Ada. This course is designed specifically for managers who will not need hands-on software know-how, the course emphasizes the role of Ada in total project development.

SYLLABUS:

- The software crisis
- Background and rationale for Ada
- What will Ada do for your organization?
- Ada transition issues
- Current status of the Ada language
- What to expect in the future

PREREQUISITES: None

RECOMMENDED FOR:

- Senior Engineering Managers
- Project Administrative Managers
- Support Managers
- System Integration Managers

NOTE: It is recommended to take this course in conjunction with Software Engineering For Managers (M101). This course module serves as a prerequisite for Ada For Software Managers (L201)

NAME: Ada Technical Overview (L102)

DURATION: 1 day

OBJECTIVE: To give the student an overview of the development and features of Ada. The course seeks to give software engineers, programmers, analysts and software engineering managers a conceptual understanding of Ada. In this course one will learn to:

- Understand how Ada differs from other HOLs
- Develop a feel for what constitutes a proper Ada style

SYLLABUS:

- Background and rationale for Ada
- Department of Defense language requirements
- A top-down development of an Ada program
- Ada program structure
- Ada features illustrated through examples
- Large system development
- Ada design criteria
- Summary of Ada constructs

PREREQUISITES: None

RECOMMENDED FOR:

- Programmers, Analysts, Software Designers
- Project/Task Leaders
- Design Consultants
- Configuration Management/Quality Assurance Engineers
- Real-Time System Architects
- System Integration Senior Technical Staff
- System Integration Engineers

NOTE: This course module is a recommended prerequisite for Basic Ada Programming (L202) and is a prerequisite for Programming Methodology (M203)

NAME: Introduction To Ada - A Higher Order Language (L103)

DURATION: 1 day

OBJECTIVE: To introduce assembler language programmers to the concept of high order languages and to the Ada Language. Programming in Ada is in many ways different from programming in assembler. This course is intended to introduce programmers, system analysts and software managers to the differences between assembly language programming and Ada programming with emphasis on the necessary shift in mind set, without which programming in Ada becomes a frustrating experience.

SYLLABUS:

- What is a high order language (HOL)?
- Pros and cons of high order languages
  - readability
  - portability
  - reusability
  - efficiency
- The Ada machine as an abstract machine.
- Programming for the abstract machine vs. programming for hardware
- Understanding Ada by understanding the abstract machine
- Introduction to data types
- Control structures
- Programming in the large
- Implementaion of the language

PREREQUISITES: Experience in assembly language programming

RECOMMENDED FOR:

- Programmers, Software Designers
- Real-Time System Architects
- Design Consultants
- Configuration Management/Quality Assurance Engineers
- System Integration Senior Technical Staff
- System Integration Engineers

NOTE: This course module serves as a prerequisite to Basic Ada Programming (L202) for those students lacking any HOL experience.

NAME:	Ada For Software Managers (L201)
DURATION:	3 days
OBJECTIVE:	To teach students how to develop and recognize a high quality software design in Ada. This course presents the Ada programming language in its entirety, but completely from the viewpoint of the technical software manager who will direct a software project without personally producing designs or code. The course de-emphasizes detailed rules in favor of a more conceptual view. In this course one will:

- Gain a thorough reading knowledge of Ada
- Learn to recognize proper and improper uses of all Ada constructs
- Understand the inevitable design trade-offs
- Learn to recognize the signs of a poor design; decide when it's time to intervene; decide which decisions are best left to advisors

SYLLABUS:

- Using Ada features in software design
  - strong typing
  - packages
  - subprograms
  - tasks
  - generics
  - overloading
  - exceptions
  - low-level features
- Characteristics of a good Ada design
  - readability
  - functional decomposition before performance
  - non-centralized database organization
  - modularity
  - use of low-level features
  - use of Ada constructs
  - design for reusability and portability

PREREQUISITES:

- Ada Orientation for Managers (L101) or Ada Technical Overview (L102)
- Software Engineering for Managers (M101) or Introduction to Software Engineering (M102)

RECOMMENDED FOR:

- Senior Engineering Managers
- Project/Task Leaders
- Design Consultants
- Support Managers
- Configuration Management/Quality Assurance Engineers
- System Integration Managers
- System Integration Senior Technical Staff
- System Integration Engineers

NOTE: This course serves as a prerequisite for Real-Time Concepts (L303)

NAME: Basic Ada Programming (L202)

DURATION: 5 days/10 days (SEE NOTE)

OBJECTIVE: To teach the student to write basic Ada programs. This course is aimed at giving thorough hands-on training in the effective use of Ada. This is the first of a series of three modules of increasing sophistication. In this course one will:

- Learn to implement small-to-medium sized modules or stand-alone programs
- Learn the use of Ada's modularity features to build software from reusable software components
- Gain a thorough understanding of what constitutes a proper Ada style

SYLLABUS:

- Lexical elements
- Introduction to data
- Enumeration types and control structures
- Numeric types
- Advanced features of scalar types
- Array types and iterative control structures
- Record types and variant records
- Program structure and separate compilation
- Using library units
- Access types
- Exceptions
- Input/Output
- Overview of other language features

PREREQUISITES: (SEE NOTE)

- Introduction To Ada - A Higher Order Language (L103) if student has no HOL experience

RECOMMENDED FOR:

- Programmers
- Software Designers
- Real-Time System Architects
- Design Consultants
- Configuration Management/Quality Assurance Engineers
- Anyone requiring hands-on knowledge of Ada

NOTE: The 5 day version of the course uses a lecture format and is intended only for those with a very strong background in high order programming languages (Pascal, C).

The 10 day version of the course consists of a combination of lecture and class exercises, with an emphasis on active participation by the class. Computer hands-on experiences will be provided where facilities permit.

It is recommended that Ada Technical Overview (L102) and Introduction to Software Engineering (M102) be taken prior to taking this course. The course is designed primarily for programmers with at least one year of experience and will serve as a prerequisite for Advanced Ada Topics (L305)

NAME: Real-Time Concepts (L303)

DURATION: 1 day

OBJECTIVE: To teach, at a conceptual level, approaches to real-time programming. This course is designed to give managers an understanding of issues and approaches to real-time programming using Ada. The course is for project leaders who need to understand designs.

SYLLABUS:

- Synchronous/asynchronous system design
- Ada tasking
- Interrupt handling
- Interfacing with the outside world
- Queue management
- Role of a run-time system

PREREQUISITES:

- Ada For Software Managers (L201)

RECOMMENDED FOR:

- Project/Task Leader

NAME: Advanced Ada Topics (L305)

DURATION: 5 days/10 days (SEE NOTE)

OBJECTIVE: To teach the student modern abstraction concepts and the related facilities of Ada. Intended to be a second Ada programming course, this course addresses design concepts of abstraction and information hiding ("object-oriented design") in the context of advanced programming techniques. This approach has the dual goal of introducing advanced techniques in the proper Ada context and, at the same time, introducing Ada design concepts in the context of interesting examples. In this course one will:

- Learn classic algorithms and data structures for common programming problems
- Learn, through numerous examples, how to use data abstraction effectively, including the use of Ada generic units
- Gain full working knowledge of all Ada types and control structures

SYLLABUS:

- Fundamental data and algorithm structuring features of Ada
  - packages
  - array types
  - record types
  - recursive procedure
- Basic data structure concepts
  - basic set types
  - linear lists
  - linear stacks and queues
  - linked lists and recursive types
  - linked stacks and queues
- Advanced data abstraction features of Ada
  - private and limited private types
  - overloading
  - generic units (subprograms and packages)

- Applications of data abstraction to basic data structures
    - linear stacks (generic)
    - linked lists (generic)
    - linked stacks (generic)
    - trees
- Low-level and implementation-dependent features of Ada.
- Overview of tasking concepts

PREREQUISITES:

- Basic Ada Programming (L202)

RECOMMENDED FOR:

- Software Designers
- Real-Time System Architects
- Design Consultants
- Configuration Management/Quality Assurance Engineers
- System Integration Senior Technical Staff
- System Integration Engineers

NOTE: The 5 day version of the course uses a lecture format and is intended only for those with a very strong background in Basic Ada programming (i.e. L202)

The 10 day version of the course consists of a combination of lecture and class exercises, with an emphasis on active participation by the class. Computer hands-on experiences will be provided where facilities permit.

This course serves as a prerequisite for Real-Time Systems In Ada (L401)

NAME: Real-Time Systems In Ada (L401)

DURATION: 5 days

OBJECTIVE: The most advanced Ada design course, this unit covers the concepts of concurrent programming in particular as they apply to real-time systems. In this course one will learn:

- Use of Ada tasking for the design of performance-critical real-time systems
- When to use the low-level features of Ada
- Recognizing issues affecting the performance; when not to be concerned with performance

SYLLABUS:

- Concurrent programming concepts
  - reasons for concurrency
- Ada tasking concepts
  - task types and task objects
  - task activation and termination
- Task cooperation
  - rendezvous
  - selective waits
  - avoiding deadlock
- Fundamental task designs
  - server and user tasks
  - monitors and message buffers
- Other tasking features
  - aborting tasks and exceptions in tasks
  - interrupt entries
  - priorities
- Improving performance
  - when and why to tune
  - tuning methods
  - scheduling techniques

PREREQUISITES:

- Advanced Ada Topics (L305)

RECOMMENDED FOR:

- Design Consultants
- Real-Time System Architects
- Configuration Management/Quality Assurance Engineers
- System Integration Senior Technical Staff
- System Integration Engineers

NOTE: This course uses a lecture format and is intended only for those with a background in concurrent programming.

NAME: Using the Ada Language Reference Manual (L402)

DURATION: 2 days

OBJECTIVE: To teach the student to use the Ada Language Reference Manual using mostly a workshop problem-solving approach.
The course teaches the proper use of the language reference manual as an indispensable tool for the software profession. In this course one will learn:

- When and how to use the manual
- The role of the manual in improving software portability and reliability
- The use of the manual as a source of design ideas

SYLLABUS:

- What is the Ada Language Reference Manual?
- Acquainting the student with the manual
  - syntax notation
  - language terms
  - references
  - annexes
  - appendices
- How to correctly interpret the manual
  - lexical elements
  - declarations and types
  - names and expressions
  - statements
  - subprograms
  - packages
  - visibility rules
  - tasks
  - program structure and compilation issues
  - exceptions
  - generic units
  - representation clauses and implementation-dependent features
  - input/output

PREREQUISITES:

- Advanced Ada Topics (L305)

RECOMMENDED FOR:

- Design Consultants
- Configuration Management/Quality Assurance Engineers
- Software Designers
- Technical Managers

METHODOLOGY COURSE MODULES

NAME: Software Engineering For Managers (M101)

DURATION: 1 day

OBJECTIVE: To teach managers modern software engineering concepts.

SYLLABUS:

- Background and motivation
- Software engineering goals
- Achieving software engineering goals
  - software life cycle
  - introduction to methods and tools
  - analysis methods
  - architectural design methods
  - detailed design methods
  - implementation methods
  - software management
- Software engineering and Ada

PREREQUISITE: None

RECOMMENDED FOR:

- Programmers
- Software Designers
- Real-Time System Architects
- Design Consultants
- Project/Task Leaders
- Configuration Management/Quality Assurance Engineers
- System Integration Senior Technical Staff
- System Integration Engineers

NOTE: This course serves as a prerequisite for Ada For Software Managers (L201) and for Software Engineering Methodologies (M201)

NAME: Introduction to Software Engineering (M102)

DURATION: 2 days

SYLLABUS:

- Background and motivation
- Software engineering goals
- Achieving software engineering goals
  - software life cycle
  - military standards and documentation
  - introduction to methods and tools
  - analysis methods
  - architectural design methods
  - detailed design methods
  - implementation methods
  - software management
- Software engineering and Ada

PREREQUISITE: None

RECOMMENDED FOR:

- Programmers
- Software Designers
- Real-Time System Architects
- Project/Task Leaders
- Configuration Management/Quality Assurance Engineers
- System Integration Senior Technical Staff
- System Integration Engineers

NOTE: This course module is a recommended prerequisite for Basic Ada Programming (L202) and Software Engineering Methodologies (M201)

NAME: Software Engineering Methodologies (M201)

DURATION: 5 days

OBJECTIVE: To provide a thorough understnading of software methodologies and how they may be used with Ada.

SYLLABUS:

- Principles of software engineering
- Life-cycle concept
- Survey of software methodologies
  - SADT
  - SREM
  - Entity Diagrams
  - PSL/PSA
  - Structured Systems Analysis Methods
  - Software Cost Reduction Project (SCRP) Methodology
  - Parnas and Object-Oriented Design
  - Constantine-Myers Structured Design
  - Jackson Structured Design
  - Warnier-Orr
  - Higher Order Software (HOS) Method
  - HIPO
- Architectural design metrics
- Program design languages
- Graphical detailed design methods
- Program complexity
- Program correctness
- Testing approaches

PREREQUISITES:

- Software Engineering for Managers (M101) or Introduction to Software Engineering (M102)

RECOMMENDED FOR:

- Design Consultants
- System Integration Senior Technical Staff

NAME: Programming Methodology (M203)

DURATION: 1.5 days

OBJECTIVE: To teach coding and documentation conventions, structured programming, stepwise refinement and programming style.

SYLLABUS:

- Structured programming concepts
- Basic control structures
- The structure theorem
- In-line documentation
- Producing the required documentation
- Programming style
- Stepwise refinement

PREREQUISITES: Ada Orientation for Managers (L101) or Ada Technical Overview (102)

RECOMMENDED FOR:

- Programmers
- Software Designers
- Real-Time System Architects
- Design Consultants
- Project/Task Leaders
- Configuration Management/Quality Assurance Engineers
- System Integration Senior Technical Staff
- System Integration Engineers

NOTE: This course serves as a prerequisite for Advanced Ada Topics (L305)

ADA LANGUAGE SYSTEM (ALS) COURSES

NAME: Ada Language System (ALS) User Course (E100)

DURATION: 10 days

OBJECTIVE: To train people in the use of the Ada Language System (ALS). In this course one will:

- Develop Ada programs using the ALS.
- Learn how to use all the ALS tools.
- Learn the features of the ALS database.
- Gain experience in the use of the ALS command language, the interface between users and ALS tools
- Learn how the ALS supports Configuration Management.

SYLLABUS:

- *Overview of the ALS.*
- Walk through: Developing an Ada Program Using the ALS
- Introduction to the Environment Database
- An ALS Session
- Introduction to the Command Language
- Substitutors
- Invoking Tools
- Manipulating Files and Directories
- More About Tools
- Attributes and Associations
- Variation Sets
- Manipulating Database Nodes
- Compiling Ada Programs
- Linking Ada Programs
- Exporting, Loading and Executing Ada Programs
- Debugging Ada Programs
- Assembling and Importing
- Writing Tools in Ada
- File Administration
- Configuration Management

PREREQUISITES:

- Knowledge of the VAX/VMS Editor (EDT)
- Some programming experience

RECOMMENDED FOR:

- Programmers
- Real-Time Architects
- Software Designers

NAME: Ada Language System (ALS) Administrator Course (E200)

DURATION: 5 days

OBJECTIVE: To train people as Ada Language System (ALS) Administrators. In the course one will learn:

- The appropriate roles for an ALS Administrator
- How to install an ALS
- How to authorize ALS users
- How to incrementally update the ALS
- How to backup and archive sections of the ALS database
- How to transmit information between ALS databases

SYLLABUS:

- Role of the ALS Administrator
- Components of the ALS
- System Installation
- User and Team Authorization
- Incremental Updates
- Database Administration and Maintenance

PREREQUISITES:

- ALS User Course
- Hands-on experience with a VAX computer and the VAX/VMS Operating System
- Some familiarity with one rudiments of computer software construction

RECOMMENDED FOR:

- Computer Operators
- System Programmers

ADA LANGUAGE WORKBOOKS

TITLE: Ada Primer

OVERVIEW: Ada Primer is the first of the series of three workbooks. The workbook leads the novice through the fundamentals of Ada by addressing the "Pascal subset" of Ada. This workbook may be used in conjunction with Basic Ada Programming (L202).

Ada Primer is intended for those individuals who have either a Bachelor's Degree in one of the sciences (with Computer Sciences courses included), or Associate in Arts Degree in Computer Science or a strong background in high order languages such as Pascal, C, Fortran, Jovial or CMS-2.

SECTIONS:

- Overview of the Ada Language
- Introduction To Program Units
- Lexical Elements
- Introduction To Data
- Enumeration: Types and Control Structures
- Numeric Types
- Advanced Features of Scalar Types
- Array Types and Iterative Control Structures
- Record Types
- Program Structure and Separate Compilation
- Using Library Units
- Exceptions
- Input/Output

TITLE: Advanced Ada

OVERVIEW: Advanced Ada is a follow-on to the Ada Primer workbook. This workbook discusses data structures and algorithms, data abstraction and information hiding. The Advanced Ada Workbook parallels and can be used in conjunction with Advanced Ada Topics (L305).

It is assumed the reader is familiar with all the concepts covered in the Ada Primer.

CHAPTERS:

- Review of Fundamental Ada Features
- Basic Data Structures
- Data Abstraction
- Applications of Data Abstraction
- Classic Applications
- Advanced Data Structures
- Implementation - Dependent Features

TITLE: Real-Time Ada

OVERVIEW: Real-Time Ada is the last workbook in the series of three workbooks. This workbook introduces concurrent programming concepts and provides exercises and solutions on selected topics in real-time systems. Real-Time Ada may be used in conjunction with Real-Time Systems In Ada (L401).

CHAPTERS:

- Concurrent Programming Concepts
- Ada Tasking Concepts
- Task Cooperation
- Fundamental Task Designs
- Special Purpose Tasking Features
- Scheduling and Optimization

SUPPLEMENTARY MATERIAL

TITLE: Case Studies Report

OVERVIEW: Case Studies Report presents a collection of case studies which illustrate in detail the effective use of Ada to solve the kinds of design problems that arise in developing embedded software systems. The case studies are of two kinds; pedagogical and observational. The pedagogical case studies present examples that can be incorporated in training material, while the observational case studies record findings that lead to a better understanding of Ada and it's usage.

CASE STUDIES:

- Power failure requirements
- Use of types to describe hardware interface requirements
- Functional description of an air defense system
- Task structure for a target tracking system
- UART: expressing hardware design in Ada
- Tasks and structure charts
- Use of dependent tasks
- Task preemption
- Queues and generics
- Stubbing and readability
- Succintness of range syntax
- Rendezvous and exit
- Decoupling partly independent activities
- Memory-mapped I/O in Ada
- Eliminating goto's
- Array of arrays

TITLE: Ada Case Studies II

OVERVIEW: Ada Case Studies II presents a set of case studies on different aspects of the Ada language. This report is a continuation of Case Studies Report. Ada Case Studies II contains fifteen case studies which illustrate different areas of the Ada language:

- naming conventions
- types
- coding paradigms
- exceptions
- program structure

These cases studies provide insight into Ada usage and style, addressing both issues that arise in embedded computer systems and general programming and design practice.

CASE STUDIES:

- Guidelines for the Selection of Identifiers
- Discrete Types
- Implementation of Set Types
- Constant Array Declarations
- Record Types
- Recursive Type Definitions
- Use of Slices
- Short Circuit Control Forms
- Loop Statements
- Use of Block Statements for Local Renaming
- The Use of Exceptions
- Specifying Interfaces for General Purpose, Portable Software: A Study of Ada Input/Output
- Information Hiding
- Reducing Depth of Nesting
- Library Units Versus Subunits

TITLE: Ada Language System (ALS) Textbook

OVERVIEW: Ada Language System (ALS) Textbook is a textbook written to teach the Ada Language System (ALS) without relying on an accompanying classroom course, though it may be used to supplement students taking Ada Language System (ALS) User's Course (E100). The textbook contains both hands-on and hands-off exercises and answers to all exercises are provided.

A small introduction to Ada is included as an appendix and no previous knowledge of Ada is required.

CHAPTERS:

- Introduction to the Ada Language System
- Walkthrough: Developing an Ada Program
- Introduction to the Environment Database
- Command Language
- Environment Database
- Compiling
- Linking
- Exporting, Loading and Executing Ada Programs
- Debugging
- Assembling and Importing
- Obtaining Information About the ALS: The Help Database
- Configuration Management Using ACC Tools

TITLE: Using Selected Features of Ada: A Collection of Papers

OVERVIEW: This report is a collection of papers written by several different authors. The purpose of these papers is to further the understanding of how to use selected features of the Ada Programming Language in a proper manner.

PAPERS:

- The Use of Ada Packages
- Types
- Tutorial on Ada Tasking
- Tutorial on Ada Exceptions
- Low Level Language Features
- Real Data Types in Ada

TITLE: Designing Real Time Systems in Ada

ABSTRACT: Real-time software differs from other kinds of software in the sense that it must interact with external events. It must detect the occurrence of certain events as soon as they happen, and exercise control over external processes in a timely fashion. Real-time software must be cheap to produce and must be extremely reliable, even more so than other kinds of software. None of the existing approaches for real-time software design have been able to satisfy all of these requirements. In this report we evaluate the role of Ada for this purpose and find that it too falls short. However Ada, unlike other approaches, can make contributions towards reducing the cost and increasing the reliability of real-time software. This report examines ideas and methods to be used in conjunction with Ada to satisfy the rest of the real-time requirements.

CHAPTERS: